<< Return to Main Page | Print

From the pages of Control Engineering

Frequency Domain Analysis Explained

Vance VanDoren -- 10/1/2005

Other useful reading

- Analyzing Control Loop Behavior in the Frequency Domain
- Assessing Control Loop Performance
- Bode plots solve frequency domain problems

AT A GLANCE

- Linear process behavior
- Fourier's Theorem and Bode plots
- Predicting future process performance
- Simple examples

Predicting the future behavior of a process is key to the analysis of feedback control systems. Knowing how the controlled process will react to the controller's efforts allows the controller to choose the course of action required to drive the process variable towards the setpoint.

Linear processes are particularly predictable since a combination of two control efforts applied simultaneously to the process will produce a process variable equal to the sum of the outputs that would have resulted had the two control efforts been applied separately. A linear process also demonstrates a constant, *steady-state gain* K. That is, if B is the value of the process variable when the control effort is zero, then the process variable eventually will settle out at a value of Y=KX+B when the control effort is fixed at a value of X. This relationship yields a straight line when Y is plotted against X (hence the expression 'linear process').

Linear processes also respond to non-constant inputs in predictable ways. Most importantly, sinusoidal inputs always yield sinusoidal outputs. In fact, if the input from the controller happens to be a sine wave with frequency v, then the process variable output by the process also will be a sine wave with the same frequency. Although real-life controllers rarely generate purely sinusoidal control efforts, this phenomenon is the basis for *frequency domain analysis* of a process' behavior.

Linear process example

A simple example of frequency domain analysis can be demonstrated by means of the child's toy shown in 'A simple linear process' graphic. This linear process consists of a weight hanging from a handle-mounted spring. A child controls the position of the weight by moving the handle up and down.

Anyone who has ever played with such a toy knows that if the handle is moved in a more-or-less sinusoidal manner, the weight will start oscillating at the same rate, though out of synch with the handle. Only at relatively low frequencies where the spring doesn't stretch will the handle and weight move in lock step.

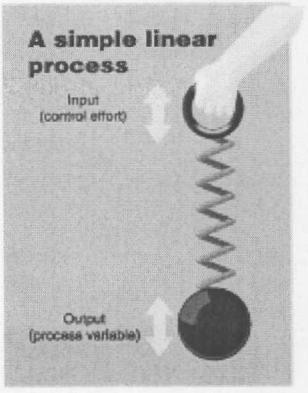
At higher and higher frequencies, the weight will begin to oscillate more than the handle yet lag further and further behind it. At the *natural frequency* of the process, the weight's oscillations will reach their maximum height. The natural frequency is determined by the mass of the weight and the stiffness of the spring.

Above the natural frequency, the *amplitude* of the weight's oscillations will decrease and its *phase* will grow more negative (that is, oscillations will grow smaller and smaller and lag further behind). At very high frequencies, the weight will move only slightly, in exactly the opposite direction of the handle.

Bode plots

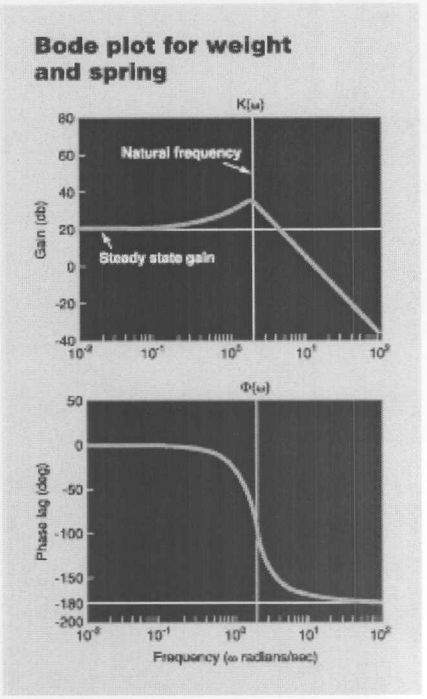
All linear processes demonstrate similar behavior. They transform a sinusoidal input into a sinusoidal

output with the same frequency but a different amplitude and phase. Just how much the amplitude and phase change depends on the *gain* and *phase lag* of the process. Gain is the factor by which the process amplifies the sine wave en route from input to output, and the phase lag is the degree by which the sine wave is delayed.



A toy comprised of a weight attached to a handle-mounted spring can illustrate frequency domain analysis. If the handle is moved in a more-or-less sinusoidal manner, the weight will oscillate at the same rate, though out of synch with the handle.

Unlike the steady-state gain K, the gain and phase lag of the process vary depending on the frequency of the incoming sine wave. The weight-and-spring process does not change the amplitude of a low frequency sine wave much. It is said to have a *low frequency gain* of one. Near the natural frequency, the gain is greater than one since the amplitude of the output is greater than the amplitude of the input. The *high frequency gain* of the process is almost zero since the weight barely oscillates at all when the toy is shaken rapidly.



A Bode plot shows how a sine wave of frequency v radians per second passing through a linear process will change its amplitude by a factor of K(v) and lose phase by F (v) degrees. K(v) and v are generally graphed on logarithmic scales. Bode plots vary in shape for different processes, but K(v) always approaches the steady-state gain as v tends to zero. For very high frequencies, K(v) generally tends towards zero. A Bode plot can be derived empirically by exercising the process with a sinusoidal control effort at various frequencies or by analyzing the physical characteristics of the process, such as the stiffness of the spring and the mass of the weight in the example process.

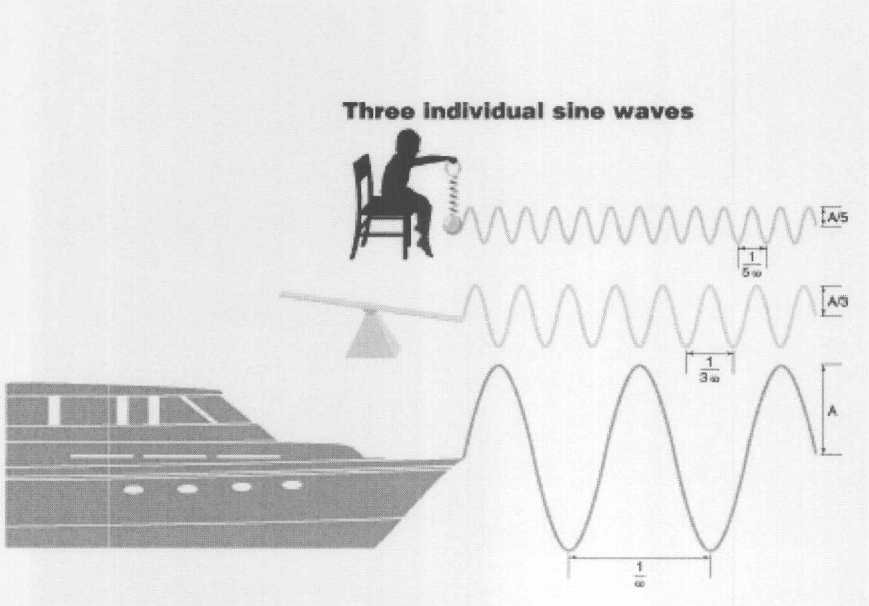
The process's phase lag is an additive factor. In this example, it starts near zero for low frequency inputs since the weight and the handle oscillate in synch when the handle is moved very slowly. The phase lag drops to -180 degrees at high frequencies where the handle and weight move in opposite directions (hence the expression '180 degrees out of phase' used to describe any situation involving complete opposites).

'Bode plot' graphic shows the complete spectrum of gains and phase lags for the weight-and-spring process at all frequencies between 0.01 and 100 radians per second. This is an example of a *Bode plot*, a graphical analysis tool developed by Hendrick Bode at Bell Labs in the 1940s. It can be used to determine the amplitude and phase of the output that results when the process is driven by a sinusoidal input with a particular frequency. To get the output amplitude, simply multiply the input amplitude by the gain shown at that frequency. To get the output phase, add the phase lag to the input phase.

Fourier's Theorem

Gains and phase lags shown in a process's Bode plot are its defining characteristics. They tell an experienced control engineer everything he needs to know about the behavior of the process and how it will respond in the future not only to sinusoidal control efforts but to *any* control effort.

Such an analysis is made possible by *Fourier's Theorem*, which states that any continuous sequence of measurements or *signal* can be expressed as an infinite sum of sine waves. Mathematician Joseph Fourier proved his famous theorem in 1822 and produced an algorithm known as the *Fourier Transform* for computing the frequency, amplitude, and phase of each sinusoid in that sum from measurements of the original signal.



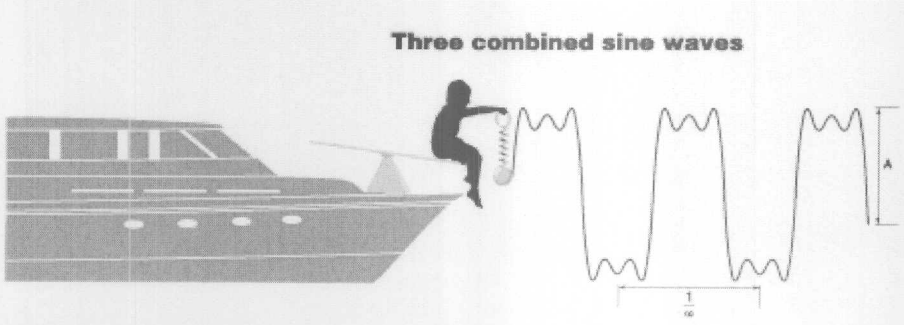
A ship oscillating at a frequency of v and an amplitude of A, a teeter-totter oscillating at a frequency of 3v and an amplitude of A/3, and a child's toy oscillating at a frequency of 5v and an amplitude of A/5 would each generate a sine wave if their motions were plotted on separate trend charts.

Theoretically, Fourier Transforms and Bode plots can be used together to predict how a linear process would react to a proposed sequence of control efforts. Here's how:

1) Use the Fourier Transform to mathematically decompose the proposed control effort into its theoretical sine wave components or *frequency spectrum*.

2) Use the Bode plot to determine how each of those sine waves would have been modified had it been passed through the process by itself. That is, apply the appropriate amplitude and phase changes dictated by each sine wave's frequency.

3) Use an inverse Fourier Transform to recombine the modified sine waves into one signal.



The combined motion of the toy, the teeter-totter, and the ship yields a square wave with a period

(inverse frequency) of 1/v and amplitude just shy of A.

Since the inverse Fourier Transform is essentially an addition operation, the linearity of the process will guarantee that the combined effect of the theoretical sine waves computed in step 1 will be the same as if they had remained summed together. Thus, the combined signal computed in step 3 will represent the process variable that would have resulted had the proposed control efforts been input to the process.

Note that at no point in this procedure are any individual sine waves generated by the controller nor plotted on paper. All such frequency domain analysis techniques are conceptual. It is a matter of mathematical convenience to translate signals from the *time domain* into the *frequency domain* with the Fourier Transform (or the closely related *Laplace Transform*), solve the problem at hand using Bode plots and other frequency domain analysis tools, then transform results back into the time domain.

Most control-design problems that can be solved in this manner can also be solved by direct manipulations in the time domain, but the calculations are generally easier in the frequency domain. In the above example, it was a matter of multiplication and subtraction to compute the frequency spectrum of the process variable given the Fourier Transform of the proposed control efforts and the Bode plot of the process.

It is not altogether obvious that adding up the right combination of sine waves will yield a signal with a desired shape as Fourier posited. An example may help.

Consider again the child's spring/weight toy, a playground teeter-totter, and a ship on the open ocean. Suppose that the ship happens to be rising and falling on the waves in a sinusoidal manner at a frequency of ù and amplitude of A. Also suppose that the teeter-totter is oscillating at three times that frequency and one-third that amplitude while a child bounces the toy at five times that frequency and one-fifth that amplitude. 'Three individual sine waves' graphic shows what those three sinusoidal motions would look like if observed separately.

Now suppose that the child is sitting on the end of the teeter-totter, which in turn is bolted to the deck of the ship. If the three individual sine waves happen to line up just right, the toy's total movement would approximate a square wave as shown in the 'Three combined sine waves' graphic.

This isn't exactly a practical example, but it does demonstrate that the sum of a base sine wave plus one-third of its *third harmonic* plus one-fifth of its *fifth harmonic* approximates a square wave with a frequency of v and an amplitude just shy of A. The approximation gets even better when one-seventh of the seventh harmonic is added plus one-ninth of the ninth harmonic. In fact, Fourier's Theorem shows that if such a summation were to be continued *ad infinitum*, the total would match a square wave of amplitude A *exactly*. Fourier's Theorem can also be used to decompose a non-periodic signal into an infinite sum of sine waves.

<< Return to Main Page | Print

<< Return to Main Page | Print

From the pages of Control Engineering

Analyzing Control Loop Behavior in the Frequency Domain

Vance VanDoren -- 7/1/2002

The *frequency domain* is a mathematical construct that simplifies the analysis of a control system's performance. It can be used to show how a process operating under the influence of a feedback controller will react to inputs from the controller or a change in the behavior of the process.

KEY WORDS

- Process and advanced control
- Control theory
- Process control systems
- Process control theory
- Tutorial

Frequency domain analysis rests on two fundamental principles. The first is *linearity,* which states that the sum of two signals applied to a linear process will produce an output equal to the sum of the outputs that would have resulted had the two inputs been applied separately. It follows that a linear process will produce a **Y** % change in the process variable (the process output) following an **X** % change in the control action (the process input) according to a straight line relationship **Y = KX** . The *steady state gain* **K** will remain constant whether the process is currently running at maximum capacity, minimum capacity, or somewhere in between.

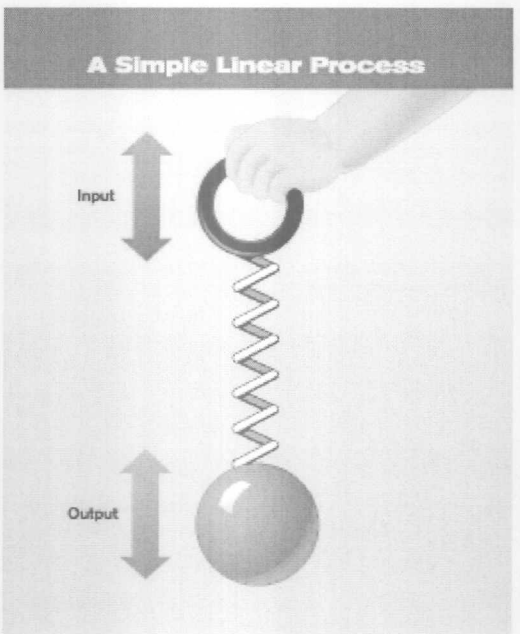
A linear process behaves somewhat differently if the input applied by the controller is not just a simple step change of **X** %. If the control action continues to oscillate sinusoidally so that the process variable never has a chance to reach a steady state, the apparent gain of the process will generally be less than **K** . That is, the amplitude of the sine wave coming out of the process will be less than **K** times the amplitude of the sine wave going in.

An example

The apparent process gain **K(ù)** will depend on the frequency ù of the incoming sine wave. Higher frequency sine waves are generally attenuated more severely than lower frequency sine waves so lower gains result at higher frequencies. This phenomenon can be demonstrated with a common child's toy comprised of a weight hung from a handle by way of a vertical spring. If the child raises and lowers the handle more-or-less sinusoidally, the weight will eventually start oscillating at the same rate as the handle. However, as the child pumps the handle faster and faster, the amplitude of the weight's oscillations will decrease until the weight no longer moves at all.

The weight's oscillations will also tend to lag behind those of the handle by an ever-increasing margin as the frequency increases. This is another phenomenon shared by all linear processes. The *phase* of the output lags the phase of the input by an amount that usually increases with frequency. Note, however, that the frequency of the incoming sine wave remains unchanged as it passes through the process. No matter how fast or slow the child pumps the handle, the weight will always oscillate at the same rate, albeit with a different amplitude and phase.

On the other hand, those amplitude and phase changes will be the same every time that a sine wave of the same frequency passes through the process. Thus, amplitude and phase changes can be plotted vs. frequency to produce two fixed curves that are characteristic of the process. These curves comprise the process's *Bode plot* , a graphical analysis tool



The process consists of a handle, a spring, and a weight. A child holding the handle serves as the controller. The

developed by Hendrick Bode (rhymes with 'roadie') at Bell Labs in the 1940s. A Bode plot can be derived empirically by exercising the process with a sinusoidal control action at various frequencies or by analyzing the physical characteristics of the process.

process output is the vertical position of the weight as measured by the child's observations. By moving the handle, the child can input a control action to change the weight's position.

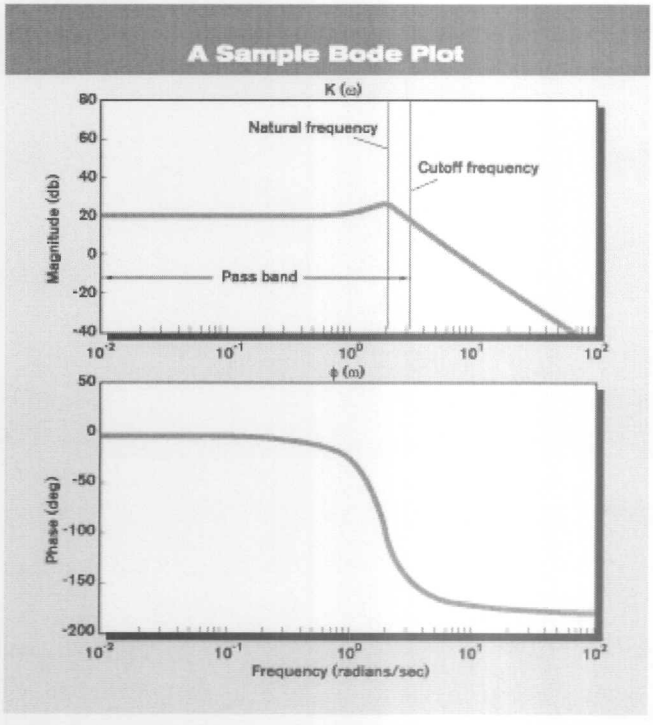
Fourier's Theorem

Of course most real-life processes are not controlled by driving them with strictly sinusoidal inputs. However, the second fundamental principle of frequency domain analysis-*Fourier's Theorem*- states that *any* signal (including a nonoscillatory control action) can be expressed as a sum of sine waves. Mathematician Joseph Fourier proved his famous theorem in 1822 and produced an algorithm known as the *Fourier Transform* for computing the frequency, amplitude, and phase of each sinusoid in that sum from measurements of the original signal.

Theoretically, then, Fourier Transforms and Bode plots can be used together to predict how a linear process would react to a proposed sequence of control actions. Here's how:

- Step 1-Use the Fourier Transform to mathematically decompose the proposed control action into its theoretical sine wave components or*frequency spectrum* .
- Step 2-Use the Bode plot to determine how each of those sine waves would have been modified had it actually been passed through the process. That is, apply the appropriate amplitude and phase changes dictated by each sine wave's frequency.
- Step 3-Use an Inverse Fourier Transform to recombine the modified sine waves back into a single signal.

Since the Inverse Fourier Transform is essentially an addition operation, the linearity of the process will guarantee that the combined effect of the theoretical sine waves computed in step 1 will be the same as if they had remained summed together. Thus, the combined signal computed in step 3 will represent the process variable that would have resulted had the proposed control action actually been input to the process.



*A Bode plot shows how a sine wave of frequency **v** radians/sec passing through a linear process will change its amplitude (or magnitude) by a factor of **K(v)** decibels and lose phase by **F(v)** degrees. **K(v)** and **v** are generally graphed on logarithmic scales. Bode plots vary in shape for different processes, but **K(v)** always approaches the steady state gain **K** as **v** tends to zero. For very high frequencies, **K (v)** tends towards zero. At some point in between known as the cutoff frequency, the magnitude plot drops to 70.7% of the steady state gain. The frequencies below this point lie within the pass band of the process. The larger the pass band or bandwidth, the more sine waves that can pass through the*

process with less than 70.7% attenuation. Bandwidth is a common measure of a communication process's capacity for transmitting data in the form of simultaneous sinusoidal signals.

Note that at no point in this procedure were any individual sine waves actually generated by the controller nor even plotted on paper. All such frequency domain analysis techniques are strictly conceptual. It is merely a matter of mathematical convenience to translate signals from the *time domain* into the *frequency domain* with the Fourier Transform (or the closely related *Laplace Transform*), solve the problem at hand using Bode plots and other frequency domain analysis tools, then transform the results back into the time domain.

Most control design problems that can be solved in this manner can also be solved by direct manipulations in the time domain, but the calculations are generally much easier in the frequency domain. In the above example, it was merely a matter of multiplication and subtraction to compute the frequency spectrum of the process variable given the Fourier Transform of the proposed control action and the Bode plot of the process.

More Bode plots

Unfortunately, there's a price to be paid for such computational convenience. It's not always easy to determine just what mathematical problem needs to be solved in the frequency domain in order to solve the original problem in the time domain.

Stability analysis is a good example. A control engineer experienced with frequency domain analysis can look at a process's Bode plot and divine from its shape just how aggressive a controller can afford to be without driving the closed-loop system unstable. And once a controller has been designed and applied to the process, the combined Bode plot for the controller and the process operating in series shows whether the closed-loop system would become either more or less stable (and by how much) if the behavior of the process were to suddenly change. These issues are both related to the *gain margin* and *phase margin* of the controller-and-process combination, but just what these concepts mean and how they can be read from a Bode plot requires a fairly extensive background in control theory.

On the other hand, *resonance analysis* is fairly straightforward. Consider the sample Bode plot shown earlier, for example. The magnitude plot shows a distinct peak at the *natural frequency* . A process with this Bode plot would amplify rather than attenuate an incoming sine wave oscillating at this particular frequency. Any process capable of storing energy will demonstrate this phenomenon known as *resonance* . If the resonant peak is high enough, the process can actually destroy itself when driven by a sine wave of just the right frequency. This is why soldiers break step when crossing bridges. If the frequency of their marching cadence happens to match the natural frequency of the bridge, it could be forced to oscillate to the point of collapsing.

This sample Bode plot could also represent the spring toy. It has a natural frequency that depends on the mass of the weight, the spring constant, and the friction in the spring. Pumping the handle at this particular frequency will cause the weight to bounce wildly (which is exactly the point of such a toy).

Processing operatiing in series

Frequency domain analysis also affords a simple solution to the problem of how multiple processes will behave when connected in series. Consider, for example, a combined process where the handle of a second spring toy has been hung from the weight of the first. The output of process *A* (the position of weight *A*) is now the input to process *B* (the position of handle *B*).

Thus, a sinusoidal input of frequency **ù** applied to process *A* would generate a sinusoidal input to process *B* of the same frequency. Process *A* would attenuate that sine wave by a factor of **K** A**(ù)** and process *B* would further attenuate it by a factor of **K** B**(ù),** for a total attenuation of **K** A**(ù)** times **K** B**(ù).** The total phase lag introduced by the two process would be additive; i.e. **Ö** A**(ù)** plus**Ö** B**(ù).** Therefore, the Bode plot of the overall system would simply be the product of the two process's magnitude plots and the sum of the two process's phase plots. Furthermore, since magnitude plots are

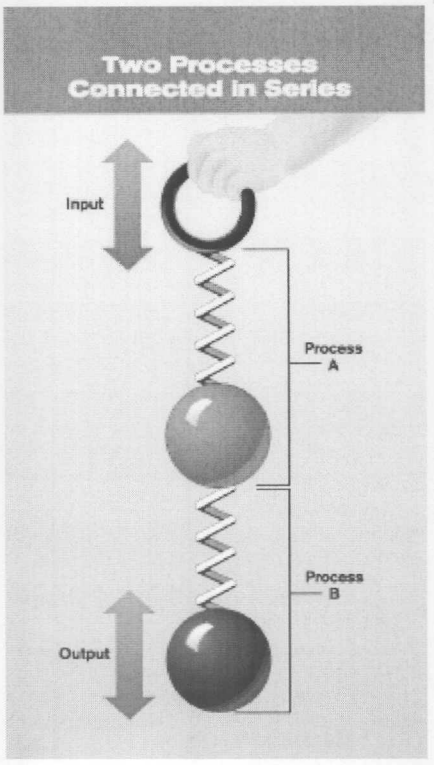
generally graphed on a logarithmic scale, the overall magnitude plot can be deduced graphically simply by adding the two individual magnitude plots together point by point.

This can be a particularly powerful analysis tool, especially when attempting to determine the overall behavior of a process operating in series with a controller. Summing the Bode plot of the controller with the Bode plot of the process yields the Bode plot of the combined system. The shape of that combined Bode plot in turn reveals much about the behavior of the closed-loop system.

Unfortunately, the whole technique breaks down if either the process or the controller is not truly linear. Controllers can generally be designed to behave linearly (and almost always are), but considerable care must be taken to ascertain the linearity of the process before frequency domain analysis can even be attempted.

For additional reading, see www.controleng.com/tutorials.

Comments? E-mail Vance VanDoren at controleng@msn.com.



The output of process A is the input to process B.

<< Return to Main Page | Print

<< Return to Main Page | Print

From the pages of Control Engineering

Assessing Control Loop Performance

Vance J. VanDoren, Control Engineering -- 5/1/1999

key words

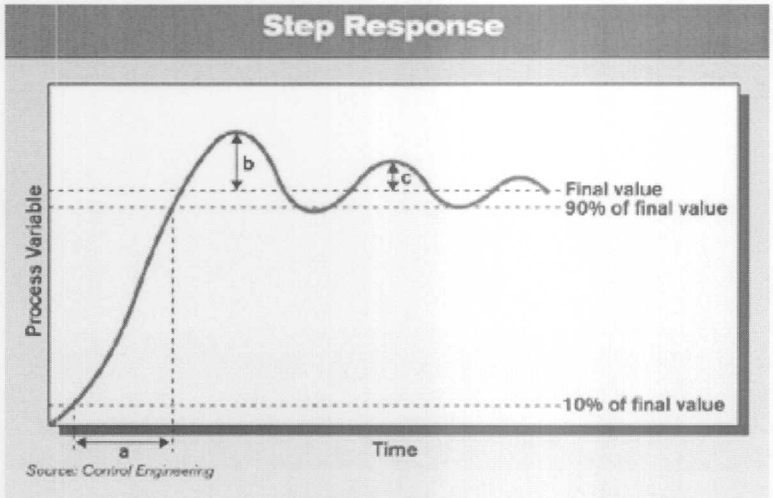
- Process control and instrumentation
- Loop tuning software
- Proportional/Integral/Derivative
- Process controllers

The basic measure of a control loop's performance is the error between the process variable and its setpoint. Zero error indicates the controller's corrective efforts have been successful in forcing the process variable to match the setpoint.

However, size of the error at any given instant is not necessarily a good indication of how well the controller is performing. Some processes are naturally quiescent. Disturbances to the process variable and changes to the setpoint are uncommon in such cases, so the error remains zero for extended periods with little or no help from the controller.

Consider, for example, a controller that regulates the water level in a large reservoir. For the most part, the controller only needs to make minor corrections to the outflow through the dam to counteract effects of evaporation and precipitation upstream. Most any controller equipped with even the simplest error feedback mechanism could handle that job.

The real test of this controller's performance comes when a flash flood hits, causing the water level to rise precipitously. If the controller takes too long to compensate with an increased outflow, the reservoir may overflow. If it reacts too quickly, it may flood the valley below. The ideal controller for this application would perform somewhere between those two extremes.



This strip chart shows how a typical PID loop forces the process variable to react when the setpoint is changed. A similar pattern may result from a disturbance, but the process variable usually ends up close to where it started since feedback controllers are generally designed to reject disturbances entirely. The rise time is a, the peak overshoot is b, and the decay ratio is c/b.

Step tests

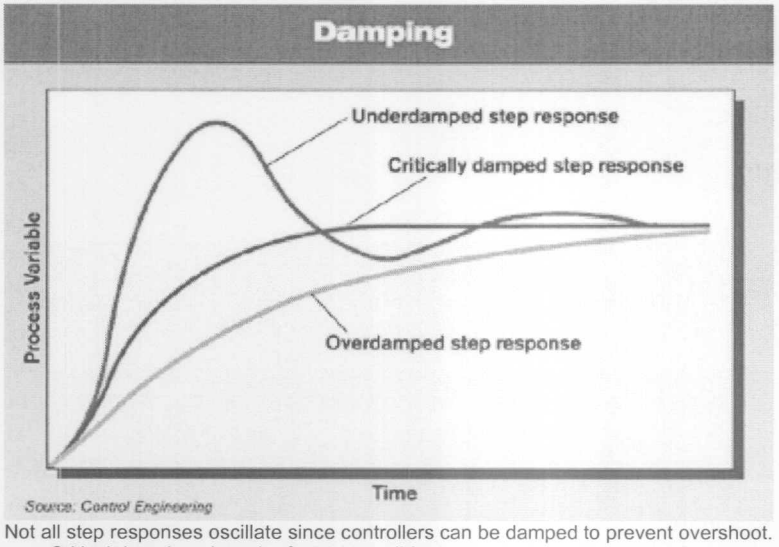
This example illustrates a more useful means of assessing a controller's actual performance level-the step test. A step test demonstrates how the controller reacts to an abrupt change in the setpoint or an abrupt disturbance to the process variable (such as a flash flood). Many PID controllers will react by forcing the process variable to oscillate as shown in the 'Step Response' strip chart. The magnitude, frequency, and duration of the oscillations in this step response are commonly used to measure the controller's performance.

For example, decay ratio can be used to quantify how quickly the controller eliminates the process variable's oscillations after the initial step is applied. It is computed by dividing the magnitude of the second peak by the magnitude of the first. Referring to the 'Step Response' chart, a low decay ratio indicates the controller can bring the process back into a steady-state or line-out condition quickly. John Ziegler and Nathaniel Nichols determined that a

quarter wave (or 25%) decay ratio was low enough for their purposes and devised their famous PID tuning rules to achieve roughly that level of performance (See 'Ziegler-Nichols Methods Facilitate Loop Tuning,' *Control Engineering*, Aug. '98)

A lower decay ratio typically is achieved at the expense of a longer rise time during which the process variable rises towards its first peak. A long rise time indicates the controller is not particularly aggressive about responding to a disturbance or a setpoint change. On the other hand, the slow rise of the process variable reduces overshoot and minimizes the process variable's subsequent oscillations. The best controller for the reservoir would have to be reasonably fast, but not too oscillatory.

In fact, the reservoir's controller should probably be designed to avoid oscillations altogether since the valley downstream would be alternately drenched and dry if the controller were to allow excess flood water to exit the dam in surges. The ideal performance criteria for applications that cannot endure oscillatory behavior is critical damping. To achieve a critically damped step response, the controller must react to an error as fast as possible without causing the process variable to overshoot the setpoint. A response that is any slower than this is said to be overdamped while a response that is any faster (and therefore oscillatory) is said to be underdamped, see 'Damping' graph.

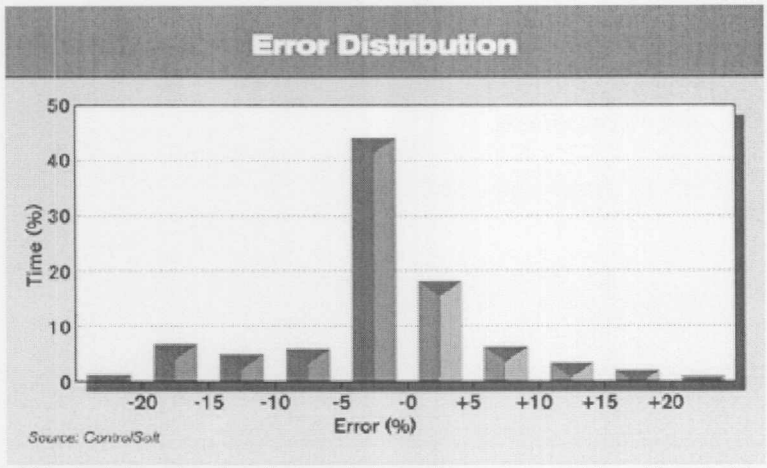


Not all step responses oscillate since controllers can be damped to prevent overshoot. Critical damping gives the fastest possible step response without overshoot.

Indirect performance measures

Step tests can reveal a great deal about a controller's performance, but they are not always practical. A control engineer probably wouldn't be allowed to flood the reservoir or abruptly change the desired water level just to see how the controller reacts.

Indirect computational techniques are often used instead. For example, ControlSoft (Cleveland, O.) has equipped the latest version of its InTune software with an error distribution chart that shows how long the error between the process variable and its setpoint has been very high positive, very high negative, almost zero, and so on; see 'Error Distribution' chart.



This chart shows how long the error between the process variable and its setpoint has remained in each of 10 ranges from 'very high negative' to 'very high positive.'

No special test inputs are required. InTune simply collects real-time error data during normal operations and displays it as a histogram. The result is a statistical view of how well the loop has been performing in the recent past. If it shows the error has been almost zero 90% of the time, then the controller has been doing its job well. If it shows the error has been dwelling in several different ranges both positive and negative, then the controller has been causing the process variable to oscillate.

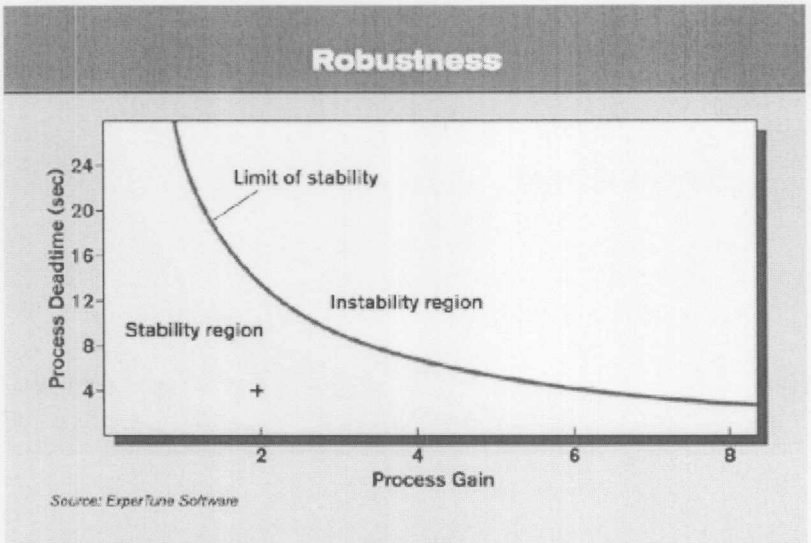
The next release of InTune will also include spectral analysis for identifying patterns in the process variable and the controller's output. Every stream of such data has its own characteristic 'spectrum,' which shows how it could be constructed from superimposed sine waves much as the spectrum of a light source shows its component frequencies. Like the spectra of similar light sources, the spectra of similar data streams show similar patterns of peaks and valleys when plotted.

Spectral analysis is particularly useful for assessing the performance of multivariable controllers that work to manipulate multiple process variables all at once. If the process variables interact, a change in one will affect all others. That can make the controller's job particularly difficult. However, a decoupling controller can be designed to coordinate its outputs so that each process variable can be manipulated more-or-less independently. The effectiveness of the decoupling effort will show up in the spectra of the process variables. Each should be different. If they show similar patterns, the decoupling has failed.

Bode plots

A related analysis tool is the Bode plot which shows how much a controller amplifies each of the sinusoidal components of the error data and how much the process amplifies each of the sinusoidal components of the controller's output. The Protuner software from Techmation (Scottsdale, Ariz.) can generate empirical Bode plots for the process and the controller from a file of historical process data. These show how well the controller and the process work together in a closed loop to amplify the desired frequencies and attenuate the unwanted frequencies.

Bode plots can also be used to assess the relative stability of the closed-loop system. If the system is unstable outright, its step response will oscillate with ever larger peaks as the controller tries to amplify the wrong frequencies too much. This makes instability easy to spot. However, a stable closed-loop system can also be on the verge of instability, which is almost as bad. What a Bode plot shows is how much more amplification the controller could afford to apply to the error signal without driving the process variable into oblivion. The Bode plot for a highly stable system shows a high gain margin; i.e., lots of room for additional amplification. For a less stable system, the Bode plot shows commensurately less gain margin. The gain margin is one of the most widely used measures of a controller's performance since it requires no special tests to compute.



This chart shows the range of first order processes that can be combined with a given PID controller to produce a stable closed-loop system. The red curve represents the controller's limit of stability which denotes the largest combinations of process gain and deadtime that this controller can handle. Combinations well within the stability region (such as the deadtime of 4 and gain of 2 shown by the cross) will be relatively more stable than combinations closer to the limit of stability. The larger the stability region, the more robust the controller.

Robustness plots

On the other hand, generating a Bode plot and reading its gain margin is not as simple as performing a step test and observing the overshoot or decay ratio. ExperTune from ExperTune Software (Hubertus, Wis.) offers a more intuitive alternative-the robustness plot. It shows the closed-loop system's relative stability by plotting the process deadtime vs. the process gain. The 'Robustness' chart shows a sample robustness plot for a process with a deadtime of 4 sec and a gain of 2 (represented by the cross on the plot).

The red curve on the robustness plot represents the influence of the controller on the stability of the closed-loop system. Any process with a deadtime and gain that falls below or to the left of the red curve will yield a stable closed loop system when combined with this particular controller. Conversely, a process with a high gain and a long deadtime would correspond to a cross above the red curve and an unstable closed loop system.

A controller that is designed to be especially conservative will yield a red curve that encompasses a wide range of dead time and gain combinations; i.e., a large stability region. Such a robust controller could be combined with most any process to create a stable closed-loop system. Conversely, a more aggressive controller would have a relatively small stability region on the robustness plot, indicating that it would be stable only with processes with low deadtime and low gain.

Tuning

The robustness plot illustrates the whole point of assessing a controller's performance-if it's too fast, too slow, too aggressive, or too conservative, re-tune it. This is generally a trial-and-error procedure. The control engineer designs a controller, tries it out, and keeps fiddling with it until the desired performance is achieved.

ACT GmbH (Tervuren, Belgium) combines those steps with a model-based controller called Topas that includes a performance-based tuning facility. Users can increase or decrease the closed-loop overshoot and the controller damping by manipulating those parameters directly. Topas automatically translates the user's specifications into the required tuning parameters. The closed-loop performance can be tuned for optimal setpoint response or for optimal disturbance rejection.

For more information about software products that can help translate performance measures into tuning parameters, see 'How Software Tools Simplify Loop Tuning,' (*CE*, Nov. '97).

For more information...

For more information, visit www.controleng.com/info:

ControlSoft **Techmation** **ExperTune Software**

ACT GmbH

<< Return to Main Page | Print

<< Return to Main Page | Print

From the pages of Control Engineering

Bode plots solve frequency domain problems

Vance J. VanDoren, consulting editor -- 5/1/1997

Every child who has ever held a spring upright knows that tugging on the top end causes the bottom end to start bouncing and that repeated tugging keeps those oscillations going. Some may notice that even though both ends always oscillate at the same frequency, the bottom end bounces higher at some frequencies than at others. Truly gifted children might even notice that the bottom end oscillates out of sync with the top end and lagsfurther and further behind as the frequency increases.

Engineers know that many mechanical, electrical, and chemical processes with energy-storing components behave the same way. A step change at the input end causes decaying oscillations at the output. A sinusoidal input, on the other hand, causes a sinusoidal output with an amplitude and a lag time that depend on the frequency of the oscillations. Not coincidentally, these two phenomena are related and form the basis of the frequency domain techniques that are fundamental to the analysis of feed-back controllers.

Two ways to calculate

There are basically two ways to analyze the behavior of a process in the frequency domain. The direct method is to drive it with a series of sinusoidal inputs, each with the same amplitude but different frequency. The amplitude and lag time of the sinusoidal outputs that result in each case can be plotted against frequency to produce a Bode plot for the process. The sample Bode plot in the figure shows how high the bottom end of the spring will bounce and how much it will lag the top end when the top end is set oscillating at various frequencies.

The second frequency domain analysis method uses Fourier's Theorem to compute the process' Bode plot indirectly. Fourier's Theorem states that any signal which is not itself a sine wave can be expressed as a sum of sine waves. A step input, for example, results when a few high-amplitude, low-frequency sine waves are added to a larger collection of low-amplitude, high-frequency sine waves. Fourier's Theorem also gives a formula for computing the amplitude of each component sine wave plus its lag time relative to the lowest frequency compo-nent. Note that plotting the amplitude and lag time for each component against its frequency yields a Bode plot of the signal just like theBode plot for an entire process.

In fact, the Bode plot for a process can be derived from the Bode plots of its input and output signals. Simply divide each amplitudein the output's Bode plot by the corresponding amplitude in the input's Bode plot. The result-ing quotient is the amplitude for the process' Bode plot at that frequency. To get the lagtimes for the process' Bode plot, simply subtract each input lag time from the correspondingoutput lag time.

Conversely, multiplying the amplitudes of an arbitrary input signal with the amplitudes of the process will give the amplitudes resulting in the output when that input is actually applied to that process. The output's lag times are computed by adding the lag times of the input to the lag times of the process. With this mathematical 'trick,' control engineers can predict the effects of a controller's actions on any process with a known Bode plot.

Consulting Editor Vance J. VanDoren, Ph.D, P.E., president of VanDoren Industries, West Lafayette, Ind.

<< Return to Main Page | Print